UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/656,654	09/05/2003	Cedric Fournet	MS1-1700US	8301

22971 7590 12/28/2007
MICROSOFT CORPORATION
ONE MICROSOFT WAY
REDMOND, WA 98052-6399

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
2191	

NOTIFICATION DATE	DELIVERY MODE
12/28/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

roks@microsoft.com
ntovar@microsoft.com
a-rydore@microsoft.com

Office Action Summary	**Application No.** 10/656,654	**Applicant(s)** FOURNET ET AL.	
	Examiner MARY STEELMAN	**Art Unit** 2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 27 August 2007.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-77 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-77 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

MARY STEELMAN
PRIMARY EXAMINER

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to Replacement Sheet Drawing (FIG. 4), Amendment to the Specification, Claims and Remarks received 08/27/2007. Per Applicant's request, claims 29-56 & 68-72 have been amended. Claims 1-77 are pending.

Claim Objections

2. In view of the amendment to Claim 51, the prior objection is hereby withdrawn.

Drawings

3. In view of the amendments to the Specification and Replacement Sheet Drawing FIG. 4, the prior objections to the drawings are hereby withdrawn.

Claim Rejections - 35 USC § 101

4. In view of the amendments to Claims 29-56 and 68-72, the prior 35 U.S.C. 101 rejections are hereby withdrawn.

35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 57-62 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Independent claim 57 is a system claim, but it lacks any hardware. The broadest reasonable interpretation is that claim limitations are directed towards software per se. Claim may be amended to recite: "A system comprising: a processing unit and memory; a call graph generator..."

Claim Rejections - 35 USC § 112

5. In view of the amendment to claim 51, the prior 35 U.S.C. 112 second paragraph rejection is hereby withdrawn.

Response to Arguments

6. Applicant's arguments filed 08/27/2007 have been fully considered but they are not persuasive.

Applicant's have argued, in substance, the following:

(A) Rioux fails to disclose (p. 20, line 1), "receiving a runtime security policy."

Examiner's Response: Examiner disagrees. See Applicant's Specification, page 6, lines 11-14, "the **assignment of rights to code** and **various security checks** performed as the code is loaded are referred to as the runtime security policy." Rioux (col. 3: 66-67), "the loader and unlinker **read ('load') the target executable code into memory** (receiving a runtime security policy) and unlink the various segments of code..." Rioux (col. 11: 3-11), "Intermediate representations of modeled executable coed can thus be **scanned or analyzed for flaws or conditions**, especially including **security holes**, buffer structure flaws exploitable via 'buffer overflow' attack, **and other known and unknown risk factors**. Such use is of great interest in the software arts today as a means of certifying software as **trusted** and/or determining whether software is safe to operate in mission-critical applications, for example."

(B) Regarding independent claims 1, 29, and 57 (and claims 63, 68, and 73 as noted on page 22, 1st paragraph), Rioux fails to disclose (p. 20, 2nd paragraph – page 21, 1st paragraph), "generating a call graph of call paths through the input component code simulated in combination with at least one symbolic component representing additional arbitrary code that complies with the runtime security policy."

Examiner's Response: Examiner disagrees. Rioux (col. 13: 12, control flow (call graph) graphs generated. Rioux (col. 14: 16-23), "The source analysis capability of source analysis project 430 provides the capability for cooperative software analysis and vulnerability (or performance) assessment. As known in the art, the term 'cooperative analysis' refers to analysis (input component code simulated) on behalf of a client who is willing to supply the original source code for a given executable program. SAF provides source code analysis through the source analysis project functions 430 shown in FIG. 3." Rioux (col. 4: 20-22), analysis (simulated) of all code- "for every procedure...every basic block...every expression...." Col. 10: 56-67, "The nanocode model resulting from the decompilation process forms the basis for (or input to) a software vulnerability or flaw analysis. In other words, the intermediate representation can be chosen so that the **model can be easily analyzed for software flaws, security vulnerability, and performance issues** (simulated...symbolic components representing additional arbitrary code that complies with the runtime security policy)...Suites of software vulnerability and other analysis tools, including scripts and automated processes, can thus be developed to operate on the IR only." Col. 12: 25-30, "Variablizer 320 comprises...a variblizer unit 322, argument detection block 324, type voting unit 326, and a simplification processor block 328. Variablizer 320 includes resource reconciliation and mapping as well as symbol interpretation and insertion. The code is run through blocks 322-328 (simulated) iteratively….until there are no more variables to process (all code is processed, including 'arbitrary code complying with the runtime security policy')." Col. 12: 41-43, "all symbol data type information is managed by symbol type manager 345, which stores its data in symbol table 347."

(C) Berg fails to disclose (page 21, 1st paragraph), "call paths through the input component code simulated in combination with at least one symbolic component representing additional arbitrary code that complies with the runtime security policy."

Examiner's Response: Rioux disclosed this limitation. See B above.

(D) Neither Rioux nor Berg disclose (page 22, 1st paragraph), "identifying a subset of the call paths in the call graph that satisfy the query."

Examiner's Response: Examiner disagrees. Rioux (col. 6: 56-60), "The CFT operates in a fashion similar to the DFT: a first, fitting control flow model is approximated from control flow graphs (subsets)…" Rioux (col. 7: 4-5), "intermediate representation used to describe..control flow model…" Rioux (col. 8: 1), "capture the fine grain detail of a nanocode level (identifiable subset of the call paths in call graph)" Rioux (col. 10: 56-58), "The nanocode model resulting…forms the basis for (or input to) a software vulnerability or flaw analysis (queries on the software)..." Rioux (col. 11: 3-10), "Intermediate representations of modeled executable code can thus be scanned or analyzed for flaws or conditions (queried) ...and unknown risk factors...certifying software as trusted and/or determining whether software is safe to operate..." Rioux (col. 10: 65-67), "Suites of software vulnerability and other analysis tools (queries to test vulnerabilities), including scripts and automated processes, can thus be developed to operate on the IR only.

More explicitly, Berg disclosed [0279] a vulnerability whereby an outside party attempts to gain privileged access to the system (query for permission to access).

Examiner maintains the rejection of claims 1-77.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

9. Claims 1-4, 27-32, and 55-58 are rejected under 35 U.S.C. 102(e) as being anticipated by US Patent 7,051,322 B2 to Rioux.

Per claims 1, 29, and 57:

A method /computer program storage medium / system comprising:

-receiving into an execution environment input component code and a runtime security policy;

-generating a call graph of call paths through the input component code simulated in combination with at least one symbolic component representing additional arbitrary code that complies with the runtime security policy (Specification: page 6, lines 11-14: the assignment of rights to code and various security checks performed as the code is loaded are referred to as the runtime security policy).

Rioux: Col. 12: 17, 22-30, symbolic representations, Variablizer comprises a variablizer unit 322, argument detection block...includes resource reconciliation and mapping as well as symbol interpretation and insertion, col. 13: 7-44, data flow graph, is passed to control flow transformer...results in a set of data and control flow graphs and associated parameters. Both the control flow and data flow of the original executable code (input component code) are completely modeled...including functions (calls), col. 13: 57-63, GUI 410, analysis generation configuration (analyze all code - additional arbitrary code), creating models for software vulnerability and / or quality assessment and related analysis and results reporting.

See Applicant's Specification, page 6, lines 11-14, "the **assignment of rights to code** and **various security checks** performed as the code is loaded are referred to as the runtime security policy." Rioux (col. 3: 66-67), "the loader and unlinker **read ('load') the target executable**

code into memory (receiving a runtime security policy) and unlink the various segments of code..." Rioux (col. 11: 3-11), "Intermediate representations of modeled executable coed can thus be **scanned or analyzed for flaws or conditions**, especially including **security holes**, buffer structure flaws exploitable via 'buffer overflow' attack, **and other known and unknown risk factors**. Such use is of great interest in the software arts today as a means of certifying software as **trusted** and/or determining whether software is safe to operate in mission-critical applications, for example."

Rioux (col. 13: 12, control flow (call graph) graphs generated. Rioux (col. 14: 16-23), "The source analysis capability of source analysis project 430 provides the capability for cooperative software analysis and vulnerability (or performance) assessment. As known in the art, the term 'cooperative analysis' refers to analysis (input component code simulated) on behalf of a client who is willing to supply the original source code for a given executable program. SAF provides source code analysis through the source analysis project functions 430 shown in FIG. 3." Rioux (col. 4: 20-22), analysis (simulated) of all code- "for every procedure…every basic block…every expression…." Col. 10: 56-67, "The nanocode model resulting from the decompilation process forms the basis for (or input to) a software vulnerability or flaw analysis. In other words, the intermediate representation can be chosen so that the **model can be easily analyzed for software flaws, security vulnerability, and performance issues** (simulated…symbolic components representing additional arbitrary code that complies with the runtime security policy)…Suites of software vulnerability and other analysis tools, including scripts and automated processes, can thus be developed to operate on the IR only." Col. 12: 25-

30, "Variablizer 320 comprises...a variblizer unit 322, argument detection block 324, type voting unit 326, and a simplification processor block 328. Variablizer 320 includes resource reconciliation and mapping as well as symbol interpretation and insertion. The code is run through blocks 322-328 (simulated) iteratively....until there are no more variables to process (all code is processed, including 'arbitrary code complying with the runtime security policy')." Col. 12: 41-43, "all symbol data type information is managed by symbol type manager 345, which stores its data in symbol table 347."

Per claims 2, 30, and 58:

-a possible execution path through the input component code that is compliant with the runtime security policy is represented by an individual call path.

Rioux: Col. 2:32-40, all paths are identified. Col. 2:41-44, analysis platform to determine if security vulnerabilities or general quality issues exist in control flow, control logic, or data organization of the modeled code.

Per claims 3 and 31:

-at least one node in the call graph includes a symbolic permission set and a known method implementation.

Rioux: Col. 10: 65-67, Suites of software vulnerability and other analysis tools, including scripts and automated processes can thus be developed to operate on the IR only. Col. 11: 3-9, Intermediate representations of modeled executable code can thus be scanned or analyzed for flows or conditions, especially including security holes, buffer structure flaws exploitable via 'buffer overflow' attack, and other known and unknown risk factors. Col. 13: 19, completely modeled, including functions (known method implementation).

Per claims 4 and 32:

-at least one node in the call graph includes a symbolic permission set and a token representing an unknown method implementation.

Rioux: Col. 11: 3-9, Intermediate representations of modeled executable code can thus be scanned or analyzed for flows or conditions, especially including security holes, buffer structure flaws exploitable via 'buffer overflow' attack, and other known and unknown risk factors.

Per claims 27 and 55:

-analyzing the call graph and another call graph obtained for a different version of input component code to generate a security report that identifies a security vulnerability in the different version of the input component code.

Rioux: Col. 2, line 42, Col. 11: 11-20 & 39, disclosed versions & reports.

Per claims 28 and 56:

-analyzing the call graph and another call graph obtained for a different version of input component code to identify a call path that presents a security vulnerability in the different version of the input component code.

Rioux: Col. 11: 11-20, disclosed testing versions.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 5-26, 33-54, and 59-77 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 7,051,322 B2 to Rioux, in view of US Patent Application 2005/0010806 A1 to Berg et al.

Per claims 5 and 33:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-initializing a symbolic value that represents data values that may be obtained by the arbitrary code at runtime.

However, Berg disclosed [0071]a 'data origin lattice 34' indicating the origin of the data, specifying that the data is internally generated relative to the analyzed routing. A symbolic value from the lattice, representing data values within an acceptable range is used to test (by the arbitrary code at runtime) the model.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 6 and 34:

Rioux failed to explicitly disclose:

-the generating operation comprises: updating a symbolic value that represents data values that may be obtained by the arbitrary code at runtime based on detection of an additional data value that may be passed as a parameter to the arbitrary code at runtime.

However, Berg disclosed [0078], a variable passed into routines as an argument.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 7 and 35:

Rioux failed to explicitly disclose:

-the generating operation comprises: updating a symbolic value that represents data values that may be obtained by the arbitrary code at runtime based on detection of a new dataflow to the arbitrary code.

However, Berg disclosed [0071], data origin lattice, data is internally generated or externally generated. [0109], expression lattice, merge results of the prior value and the expression lattice for the expression being assigned (update symbolic value).

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 8 and 36:

Rioux failed to explicitly disclose:

-the generating operation comprises:

generating a class hierarchy that contains classes of the input component code and symbolic classes that represent classes of arbitrary code.

However, Berg disclosed [0289] ANSI C language, known language using classes, such as "string class', 'array class', etc.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 9 and 37:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-identifying one or more methods of the input code component that can be called by the arbitrary code.

However, Berg disclosed [0273], as an example a call to access() a file name (input code component), and testing (called by the arbitrary code) the return value from access().

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 10 and 38:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-identifying one or more methods of the input component code that can be called by the arbitrary code;

-identifying one or more other methods of the input component code that can be called by the identified one or more methods of the input component code.

However, Berg disclosed [0264], race condition means a pair of routine calls (one or more methods / one or more other methods) that happen sequentially in a program and which, if not performed atomically, could become a vulnerability. See example code at [0266-0272].

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 11 and 39:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-identifying one or more methods of the input component code that can be called by the arbitrary code;

-identifying at least one method of the arbitrary code that can be called by a virtual call of the identified one or more methods of the input component code.

However, Berg disclosed [0274], (the calling method is virtual when the derived class's function is called) [0278], arguments to a routine may be algorithmically analyzed in view of some known behavior about the routine to detect problematic calls. Also see [0027-0028].

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 12 and 40:

Rioux failed to explicitly disclose:

-wherein any method reachable by execution in accordance with the runtime security policy is represented by one of more nodes in the call graph.

However, Berg disclosed [0025], creating an intermediate representation (IR) model. Models are used in conjunction with a vulnerability database in a vulnerability assessment to determine whether a vulnerability exists. [0027], models the arguments used to call select procedures, functions or routines.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 13 and 41:

Rioux failed to explicitly disclose:

-wherein the generating operation comprises:

-generating at least one constraint associated with one or more instructions in the input component code.

However, Berg disclosed, [0040] a memory size lattice, indicating the possible range of sizes (associated constraint) of a block of memory.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 14 and 42:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-generating at least one constraint associated with a parameter of a method call in the input component code.

However, Berg disclosed [0049], The size of the block of memory pointed to by the variable in c, in the case could be either 100 bytes or 200 bytes (generate constraint), depending on whether the array a or the array b is selected, which in turn depends on whether another variable is i or 0 (parameter of method call). (See code block at [0048].)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 15 and 43:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-generating at least one constraint associated with a returned result of a method call in the input component code.

However, Berg disclosed [0078-0079] a variable analyzed to be an array or structure, determined to be visible to other routines or passed into other routines as an argument (as a returned result), and setting the vulnerability lattice to appropriate values.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 16 and 44:

Rioux failed to explicitly disclose:

-analyzing the call graph to identify a call path that presents a security vulnerability in the input component code.

However Berg disclosed [0027-0028], The analysis applies rules to determine inherent vulnerability or risk for certain types of errors.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 17 and 45:

Rioux failed to explicitly disclose:

-analyzing the call graph to identify a call path that presents a security vulnerability in the input component code and a call path that presents no security vulnerability in the input component code.

However, Berg disclosed, [0008], [0028], determines whether a given routine call…poses an inherent vulnerability. [0221], select routine calls (determines security vulnerability or no security vulnerability).

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 18 and 46:

Rioux failed to explicitly disclose:

-analyzing the call graph to generate a security report that identifies a security vulnerability in the input component code.

However, Berg disclosed, [0225], reports.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 19 and 47:

Rioux failed to explicitly disclose:

-analyzing the call graph to identify a call path that satisfies a query.

However, as an example, Berg disclosed [0279] the vulnerability of system privileges, an outside party querying to gain privileged access to the system.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 20, 48, and 59:

Rioux failed to explicitly disclose:

-analyzing the call graph to identify a security-vulnerable usage of a permission demand. However, Berg disclosed, as an example, an outside party attempting to gain privileged access to the system [0279-0288]. The system analyzes in view of some known behavior about the routine.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 21, 49, and 60:

Rioux failed to explicitly disclose:

-analyzing the call graph to identify a security-vulnerable usage of a permission assertion.

However, Berg disclosed, as an example,[0281], analyzing the calls to identify vulnerable usage of permission assertions.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 22, 50, and 61:

Rioux failed to explicitly disclose:

-analyzing the call graph to identify a lack of uniform usage of security checks.

However, Berg disclosed [0282], resource's ACL (access control list) should never be set to null (identify a lack of uniform usage) because the resource would then be accessible or modifiable by an unauthorized user.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 23, 51, and 62:

Rioux failed to explicitly disclose:

-analyzing the call graph to identify an equivalence between use of a permission link-demand and a permission demand.

However, Berg disclosed [0285-0286], in the case of SetSecurityDespriptorDacl, an examination of the arguments to the call and knowledge about potential vulnerability, would flag a privileged access call.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 24 and 52:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-generating a class hierarchy that contains classes of the input component code and symbolic classes that represent classes of the arbitrary code;

-generating at least one constraint associated with a virtual call in the input component code;

-evaluating the at least one constraint by a symbolic computation on potential target classes for the virtual call in the generated class hierarchy.

However, Berg disclosed [0046] c / c++ language, known to use class formats. [0283], parser creates an IR, provides a symbol table, includes control flow statements. Also see rejection of claims 11, 13, and 14 above, as related to virtual calls and constraints.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 25 and 53:

Rioux failed to explicitly disclose:

-the generating operation comprises:

-generating at least one constraint associated with either a security demand or a security assert in the input component code;

-evaluating the at least one constraint by a symbolic computation on dynamic permissions of the input component code and on a parameter permission of the security demand or the security assert;

-conditionally generating at least one additional constraint associated with one or more instructions located in the input component code after the security demand or assert, responsive to the evaluating operation.

However, Berg disclosed [0222], a vulnerability database of pre-identified routines and the conditions that can cause a vulnerability. Accordingly lattices (constraints) are formed to test the code.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 26 and 54:

Rioux failed to explicitly disclose:

-analyzing the call graph to classify, based on permissions, pieces of code performing sensitive actions in the input component code.

However, Berg disclosed [0028-0029] analyzing the call graph (IL) to classify pieces of code performing sensitive actions. As an example [0279] discloses 'based on permissions.'

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 63, 68, and 73:

Rioux failed to explicitly disclose:

A method comprising:

analyzing relative to at least one query a call graph of call paths through input component code simulated in combination with at least one symbolic component representing additional arbitrary code that complies with a runtime security policy;

-identifying a subset of the call paths in the call graph that satisfy the query.

However, Berg disclosed [0276-0277] branches / arbitrary control flow (subset of call paths / additional arbitrary code). Also, see rejections addressed in claims 1 and 19 above.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Rioux, using the teachings of Berg, because Berg [0005] recognized the difficulty in detecting vulnerabilities in the programs. Likewise, Rioux recognized the need (col. 2: 9) for complete security vulnerability analyses and forensic study of failed, malfunctioning, or suspect code.

Per claims 64, 69, and 74:

See rejections of limitations as addressed in claim 59, noted above.

Per claims 65, 70, and 75:

See rejections of limitations as addressed in claim 60, noted above.

Per claims 66, 71, and 76:

See rejections of limitations as addressed in claim 61, noted above.

Per claims 67, 72, and 77:

See rejections of limitations as addressed in claim 62, noted above.

Conclusion

12. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Note Prior Art USPN 7,076,804 B2 to Kershenbaum et al.

Abstract – program graph identifies the code within the bounded paths in the program graph that use authorization.

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached at (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned: 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Mary Steelman/

MARY STEELMAN
PRIMARY EXAMINER

Primary Examiner

12/13/2007